

```
/* chloroprene.model.c for R deSolve package

Model File: chloroprene.model

Date: Thu Jul 19 13:50:54 2018

Created by: "C:/Users/JCAMPB~1.RAM/Desktop/CHLORO~1/mod/mod.exe
v5.6.5"
-- a model preprocessor by Don Maszle

Copyright (c) 1993-2015 Free Software Foundation, Inc.

Model calculations for compartmental model:

11 States:
AI = 0.0,
AX = 0.0,
AM = 0.0,
AMLU = 0.0,
AMK = 0.0,
ALU = 0.0,
AL = 0.0,
AK = 0.0,
AS = 0.0,
AR = 0.0,
AF = 0.0,

15 Outputs:
"MASBAL",
"CLU",
"CL",
"CK",
"CS",
"CR",
"CF",
"CVLUM",
"ppm",
"AMP",
"AMPLU",
"AMPK",
"cvl",
"qcbal",
"vbal",

1 Input:
EXPULSE (forcing function)

30 Parameters:
BW = 0.03,
QPC = 30.,
QCC = 30.,
QLC = 0.161,
```

```

QFC = 0.07,
QSC = 0.15,
QKC = 0.10,
VLC = 0.055,
VLUC = 0.007,
VFC = 0.05,
VRC = 0.014,
VSC = 0.77,
VKC = 0.014,
PL = 1.25,
PLU = 2.38,
PF = 17.3,
PS = 0.58,
PR = 1.76,
PB = 7.83,
PK = 1.76,
MW = 88.5,
VMAXC = 8.88,
KM = 0.08,
VMAXCLU = 0.11,
KMLU = 0.25,
KFLUC = 0.0,
VMAXCKid = 0.03,
KMKD = 9.59,
TSTOP = 7.0,
CONC = 13.0,
*/,
#include <R.h>

/* Model variables: States */
#define ID_AI 0x00000
#define ID_AX 0x00001
#define ID_AM 0x00002
#define ID_AMLU 0x00003
#define ID_AMK 0x00004
#define ID_ALU 0x00005
#define ID_AL 0x00006
#define ID_AK 0x00007
#define ID_AS 0x00008
#define ID_AR 0x00009
#define ID_AF 0x0000a

/* Model variables: Outputs */
#define ID_MASBAL 0x00000
#define ID_CLU 0x00001
#define ID_CL 0x00002
#define ID_CK 0x00003
#define ID_CS 0x00004
#define ID_CR 0x00005
#define ID_CF 0x00006
#define ID_CVLUM 0x00007
#define ID_ppm 0x00008
#define ID_AMP 0x00009

```

```

#define ID_AMPLU 0x0000a
#define ID_AMPK 0x0000b
#define ID_cvl 0x0000c
#define ID_qcbal 0x0000d
#define ID_vbal 0x0000e

/* Parameters */
static double parms[30];

#define BW parms[0]
#define QPC parms[1]
#define QCC parms[2]
#define QLC parms[3]
#define QFC parms[4]
#define QSC parms[5]
#define QKC parms[6]
#define VLC parms[7]
#define VLUC parms[8]
#define VFC parms[9]
#define VRC parms[10]
#define VSC parms[11]
#define VKC parms[12]
#define PL parms[13]
#define PLU parms[14]
#define PF parms[15]
#define PS parms[16]
#define PR parms[17]
#define PB parms[18]
#define PK parms[19]
#define MW parms[20]
#define VMAXC parms[21]
#define KM parms[22]
#define VMAXCLU parms[23]
#define KMLU parms[24]
#define KFLUC parms[25]
#define VMAXCKid parms[26]
#define KMKD parms[27]
#define TSTOP parms[28]
#define CONC parms[29]

/* Forcing (Input) functions */
static double forc[1];

#define EXPPULSE forc[0]

/*----- Initializers */
void initmod (void (*odeparms)(int *, double *))
{
    int N=30;
    odepsms(&N, parms);
}

void initforc (void (*odeforcs)(int *, double *))
{

```

```

    int N=1;
    odeforcs (&N, forc);
}

void getParms (double *inParms, double *out, int *nout) {
/*---- Model scaling */

    int i;

    for (i = 0; i < *nout; i++) {
        parms[i] = inParms[i];
    }

    for (i = 0; i < *nout; i++) {
        out[i] = parms[i];
    }
}

/*---- Dynamics section */

void derivs (int *neq, double *pdTime, double *y, double *ydot, double
*yout, int *ip)
{
    /* local */ double QC;
    /* local */ double QP;
    /* local */ double QL;
    /* local */ double QF;
    /* local */ double QS;
    /* local */ double QK;
    /* local */ double QRC;
    /* local */ double QR;
    /* local */ double VL;
    /* local */ double VLU;
    /* local */ double VF;
    /* local */ double VS;
    /* local */ double VR;
    /* local */ double VK;
    /* local */ double ROBC;
    /* local */ double VMAX;
    /* local */ double VMAXLU;
    /* local */ double KFLU;
    /* local */ double VMAXKD;
    /* local */ double CIX;
    /* local */ double CI;
    /* local */ double CVLU;
    /* local */ double CVL;
    /* local */ double CVK;
    /* local */ double CVS;
    /* local */ double CVR;
    /* local */ double CVF;
    /* local */ double CPU;
    /* local */ double CX;
    /* local */ double CV;
}

```

```

/* local */ double CPUM;
/* local */ double RAI;
/* local */ double RAX;
/* local */ double RAM;
/* local */ double RAMLU;
/* local */ double RAMK;
/* local */ double RALU;
/* local */ double RAL;
/* local */ double RAK;
/* local */ double RAS;
/* local */ double RAR;
/* local */ double RAF;

QC = QCC * pow ( BW , 0.75 ) ;

QP = QPC * pow ( BW , 0.75 ) ;

QL = QLC * QC ;

QF = QFC * QC ;

QS = QSC * QC ;

QK = QKC * QC ;

QRC = 1 - QLC - QKC - QFC - QSC ;

QR = QRC * QC ;

VL = VLC * BW ;

VLU = VLUC * BW ;

VF = VFC * BW ;

VS = VSC * BW ;

VR = VRC * BW ;

VK = VKC * BW ;

ROBC = 1 - VLC - VLUC - VFC - VSC - VRC - VKC ;

VMAX = VMAXC * pow ( BW , 0.75 ) ;

VMAXLU = VMAXCLU * pow ( BW , 0.75 ) ;

KFLU = KFLUC * BW ;

VMAXKD = VMAXCKid * pow ( BW , 0.75 ) ;

CIX = CONC * MW / 24450 ;

CI = CIX * EXPPULSE ;

```

```

CVLU = y[ID_ALU] / ( VLU * PLU ) ;

CVL = y[ID_AL] / ( VL * PL ) ;

CVK = y[ID_AK] / ( VK * PK ) ;

CVS = y[ID_AS] / ( VS * PS ) ;

CVR = y[ID_AR] / ( VR * PR ) ;

CVF = y[ID_AF] / ( VF * PF ) ;

CPU = ( QP * CI + ( QF * CVF + QL * CVL + QS * CVS + QR * CVR + QK * CVK ) ) / ( QP / PB + QC ) ;

CX = CPU / PB ;

CV = ( QF * CVF + QL * CVL + QS * CVS + QR * CVR + QK * CVK ) / QC ;

CPUM = CPU * 1000 / MW ;

RAI = QP * CI ;

ydot[ID_AI] = RAI ;

RAX = QP * CX ;

ydot[ID_AX] = RAX ;

RAM = VMAX * CVL / ( KM + CVL ) ;

ydot[ID_AM] = RAM ;

RAMLU = VMAXLU * CVLU / ( KMLU + CVLU ) + KFLU * CVLU ;

ydot[ID_AMLU] = RAMLU ;

RAMK = VMAXKD * CVK / ( KMKD + CVK ) ;

ydot[ID_AMK] = RAMK ;

RALU = QC * ( CPU - CVLU ) - RAMLU ;

ydot[ID_ALU] = RALU ;

RAL = QL * ( CVLU - CVL ) - RAM ;

ydot[ID_AL] = RAL ;

RAK = QK * ( CVLU - CVK ) - RAMK ;

ydot[ID_AK] = RAK ;

```

```

RAS = QS * ( CVLU - CVS ) ;

ydot[ID_AS] = RAS ;

RAR = QR * ( CVLU - CVR ) ;

ydot[ID_AR] = RAR ;

RAF = QF * ( CVLU - CVF ) ;

ydot[ID_AF] = RAF ;

yout[ID_MASBAL] = y[ID_AI] - y[ID_AX] - ( y[ID_AL] + y[ID_AM] +
y[ID_AMLU] + y[ID_ALU] + y[ID_AK] + y[ID_AMK] + y[ID_AS] + y[ID_AR] +
y[ID_AF] ) ;

yout[ID_CLU] = y[ID_ALU] / VLU ;
yout[ID_CL] = y[ID_AL] / VL ;
yout[ID_CK] = y[ID_AK] / VK ;
yout[ID_CS] = y[ID_AS] / VS ;
yout[ID_CR] = y[ID_AR] / VR ;
yout[ID_CF] = y[ID_AF] / VF ;

yout[ID_CVLU] = CVLU * 1000 / MW ;

yout[ID_ppm] = CONC ;
yout[ID_AMP] = ( ( y[ID_AM] * 1000 / MW ) / ( VL * 1000 ) ) / ( TSTOP /
24 ) ;
yout[ID_AMPLU] = ( ( y[ID_AMLU] * 1000 / MW ) / ( VLU * 1000 ) ) / (
TSTOP / 24 ) ;
yout[ID_AMPK] = ( ( y[ID_AMK] * 1000 / MW ) / ( VK * 1000 ) ) / ( TSTOP /
24 ) ;

yout[ID_cv1] = CVL ;

yout[ID_qcbal] = QC - QL - QF - QS - QK - QR ;

yout[ID_vbal] = BW * ( 1 - ROBC ) - VL - VLU - VF - VS - VK - VR ;

} /* derivs */

/*----- Jacobian calculations: */
void jac (int *neq, double *t, double *y, int *ml, int *mu, double *pd,
int *nrowpd, double *yout, int *ip)
{

} /* jac */

/*----- Events calculations: */
void event (int *n, double *t, double *y)
{

```

```
} /* event */  
  
/*----- Roots calculations: */  
void root (int *neq, double *t, double *y, int *ng, double *gout, double  
*out, int *ip)  
{  
  
} /* root */
```